Assignment 2

Question 1

Step 0–

Answer:

```
library(fpp3)

# Use file.choose() to interactively select the CSV file
file_path <- file.choose()

# Read the selected CSV file into a data frame
data <- read.csv(file_path)
head(data)
```

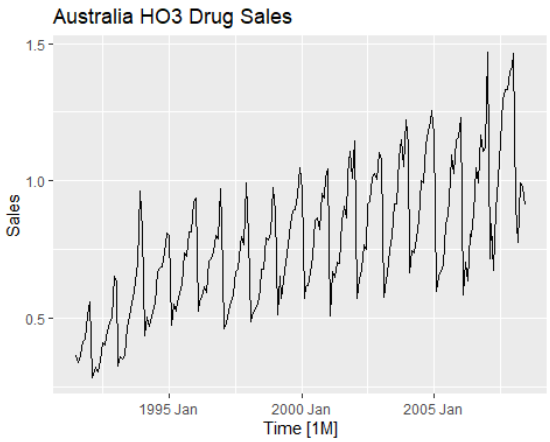
```
##   Time   Period    Sales
## 1    1 1991 Jul 365325.8
## 2    2 1991 Aug 340770.1
## 3    3 1991 Sep 367335.2
## 4    4 1991 Oct 418661.5
## 5    5 1991 Nov 427013.7
## 6    6 1991 Dec 512254.2
```

```
# Converting dataset to tsibble, and scaling the data by 1e6:
df <- data |>
  mutate(Time = yearmonth(Period),
         Sales = Sales / 1e6) |>
  as_tsibble(index = Time)
df <- df |>
  select(-Period)

head(df)
```

```
## # A tsibble: 6 x 2 [1M]
##       Time Sales
##      <mth> <dbl>
## 1 1991 Jul 0.365
## 2 1991 Aug 0.341
## 3 1991 Sep 0.367
## 4 1991 Oct 0.419
## 5 1991 Nov 0.427
## 6 1991 Dec 0.512
```

(a) Answer:

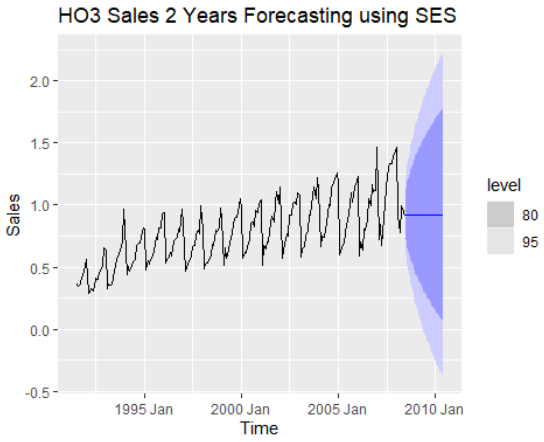
```
df |>
autoplot(Sales) +
labs(title="Australia HO3 Drug Sales",
y="Sales")
```



The provided dataset about a HO3 Drug Sales from Australia exhibits a recurring seasonal pattern, with a sharp decline occurring every January at its peak. The fluctuations in the data are closely tied to the observed Sales levels during various time intervals. Additionally, there is a distinct proportional variation (funnel like) and a gradual upward trend evident over time. Taking into account these features, it can be inferred that the time series demonstrates non-stationary behavior.

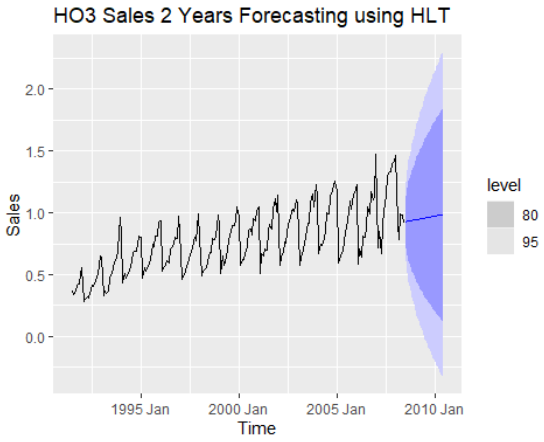
(b) Answer:

```
# Forecast the next 2 years, using Simple Exponential Smoothing (SES):
fit <- df |>
model(
SES = ETS(Sales ~ error("A") + trend("N") + season("N")),
)
fc <- fit |>
  forecast(h = 24)
fc |>
  autoplot(df) +
  labs(title="HO3 Sales 2 Years Forecasting using SES",
       y="Sales")
```

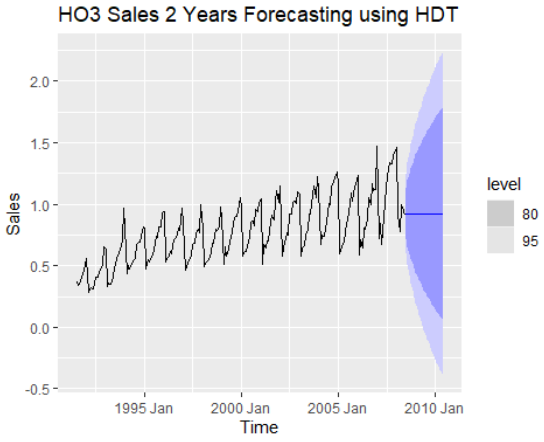



```
# Forecast the next 2 years, using Holt's linear trend (HLT):
fit <- df |>
model(
```

```
HLT = ETS(Sales ~ error("A") + trend("A") + season("N")),
)
fc <- fit |>
  forecast(h = 24)
fc |>
  autoplot(df) +
  labs(title="HO3 Sales 2 Years Forecasting using HLT",
       y="Sales")
```




```
# Forecast the next 2 years, using Holt's damped trend (HDT):
fit <- df |>
model(
HDT = ETS(Sales ~ error("A") + trend("Ad") + season("N"))
)
fc <- fit |>
  forecast(h = 24)
fc |>
  autoplot(df) +
  labs(title="HO3 Sales 2 Years Forecasting using HDT",
       y="Sales")
```

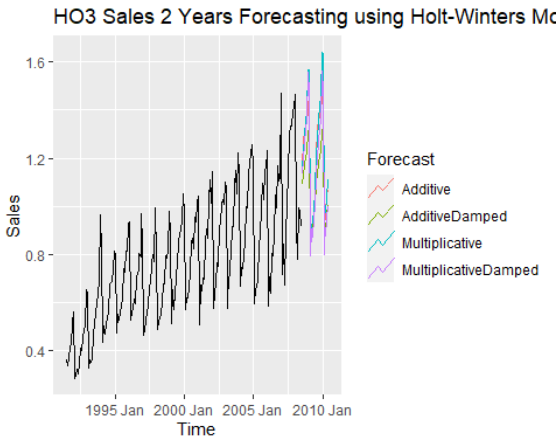


The Holt's Damped Trend (HDT) technique is employed when the Holts Linear Trend (HLT) method shows an incessant, unending increase or decrease in the trend as it extends into the future. The purpose of using HDT is to introduce a damping effect, which eventually levels the trend to a flat line at some point in the future. Upon analyzing the two-year forecast plot, a noteworthy resemblance is observed with the Simple Exponential Smoothing (SES) method, as both methods yield a consistent and unchanging prediction over time.

Hence, when considering the three methods applied to a time series featuring both trend and seasonality patterns, it appears that the methods under analysis may prove too intricate to handle these complexities.

(c) Answer:

```
# Forecasting 2 years using Holt-Winter's Seasonal methods (HW), observing
multiplicative and additive holt-winters, and with damped for both methods:
fit_hw <- df |>
  model(model(Multiplicative = ETS(Sales ~ error("M") + trend("A") + season("M")),
        Additive = ETS(Sales ~ error("A") + trend("A") + season("A")),
        MultiplicativeDamped = ETS(Sales ~ error("M") + trend("Ad") + season("M")),
        AdditiveDamped = ETS(Sales ~ error("A") + trend("Ad") + season("A")))
fc_hw <- fit_hw |>
  forecast(h = 24)
fc_hw |>
  autoplot(df, level = NULL) +
  labs(title="HO3 Sales 2 Years Forecasting using HWmul/ HWad",
       y="Sales") +
  guides(colour = guide_legend(title = "Forecast"))
```



Upon analyzing the two-year sales forecast for HO3, the Holt-Winters method, designed to account for seasonality in time series data, was applied using both additive and multiplicative methods, along with damping in both cases. The results revealed that for this seasonal data, the most accurate and robust forecasting method is the Holt-Winters model with a damped trend and multiplicative seasonality. This is attributed to the observation that the multiplicative method exhibited higher variance and increasing trend, which was mitigated by the application of damping. The additive method, followed closely by the damped additive method, showed a slight reduction in trend and variance.

The choice between these methods hinges on the nature of seasonal variation in the data. The additive method is preferable when seasonal variance remains roughly constant throughout the series, while the multiplicative method is favored when these variances change proportionally with the overall level of the series. In the case of this specific time series, it is evident that the seasonal variations exhibit a proportional

relationship with the series level. Therefore, for this time series, the Holt-Winters model with a damped trend and multiplicative seasonality emerges as the superior choice for forecasting the next two years, aligning with findings in the literature.

---

[d] Answer:

```
# Generate one-step-ahead forecasts, applying cross validation:
tr_stp1 <- df |> slice(1:(n()-1)) |>
  stretch_tsibble(.init = 36, .step = 1)
fc_stp1 <- tr_stp1 |>
  model(
    SES = ETS(Sales ~ error("A") + trend("N") + season("N")),
    HLT = ETS(Sales ~ error("A") + trend("A") + season("N")),
    HDT = ETS(Sales ~ error("A") + trend("Ad") + season("N")),
    HWmul = ETS(Sales ~ error("M") + trend("A") + season("M")),
    HWad = ETS(Sales ~ error("A") + trend("Ad") + season("A")),
    HWDmul = ETS(Sales ~ error("M") + trend("Ad") + season("M")),
    HWDad = ETS(Sales ~ error("A") + trend("Ad") + season("A"))) |>
  forecast(h = 1)
fc_stp1 |>
  accuracy(df) |>
  select(-ME, -MPE, -MAPE, -MASE, -RMSSE, -ACF1)

## # A tibble: 7 × 4
##   .model .type   RMSE    MAE
##   <chr>  <chr>  <dbl>  <dbl>
## 1 HDT    Test  0.168  0.104
## 2 HLT    Test  0.173  0.110
## 3 HWad   Test  0.0731 0.0552
## 4 HWDad  Test  0.0731 0.0552
## 5 HWDmul Test  0.0709 0.0539
## 6 HWmul  Test  0.0730 0.0555
## 7 SES    Test  0.168  0.104

# Generate four-step-ahead forecasts
tr_stp4 <- df |> slice(1:(n()-4)) |>
  stretch_tsibble(.init = 36, .step = 4)
fc_stp4 <- tr_stp1 |>
  model(
    SES = ETS(Sales ~ error("A") + trend("N") + season("N")),
    HLT = ETS(Sales ~ error("A") + trend("A") + season("N")),
    HDT = ETS(Sales ~ error("A") + trend("Ad") + season("N")),
    HWmul = ETS(Sales ~ error("M") + trend("A") + season("M")),
    HWad = ETS(Sales ~ error("A") + trend("Ad") + season("A")),
    HWDmul = ETS(Sales ~ error("M") + trend("Ad") + season("M")),
    HWDad = ETS(Sales ~ error("A") + trend("Ad") + season("A"))) |>
  forecast(h = 4)
fc_stp4 |>
  accuracy(df) |>
  select(-ME, -MPE, -MAPE, -MASE, -RMSSE, -ACF1)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be
treated as missing.
## 3 observations are missing between 2008 Jul and 2008 Sep

## # A tibble: 7 × 4
##   .model .type   RMSE    MAE
##   <chr>  <chr>  <dbl>  <dbl>
## 1 HDT    Test  0.240  0.186
## 2 HLT    Test  0.242  0.186
## 3 HWad   Test  0.0789 0.0602
## 4 HWDad  Test  0.0789 0.0602
## 5 HWDmul Test  0.0782 0.0601
## 6 HWmul  Test  0.0787 0.0606
## 7 SES    Test  0.240  0.186
```

```
# Generate six-step-ahead forecasts
tr_stp6 <- df |> slice(1:(n()-6)) |>
  stretch_tsibble(.init = 36, .step = 6)
fc_stp6 <- tr_stp1 |>
  model(
    SES = ETS(Sales ~ error("A") + trend("N") + season("N")),
    HLT = ETS(Sales ~ error("A") + trend("A") + season("N")),
    HDT = ETS(Sales ~ error("A") + trend("Ad") + season("N")),
    HWmul = ETS(Sales ~ error("M") + trend("A") + season("M")),
    HWad = ETS(Sales ~ error("A") + trend("Ad") + season("A")),
    HWDmul = ETS(Sales ~ error("M") + trend("Ad") + season("M")),
    HWDad = ETS(Sales ~ error("A") + trend("Ad") + season("A"))) |>
  forecast(h = 6)
fc_stp6 |>
  accuracy(df) |>
  select(-ME, -MPE, -MAPE, -MASE, -RMSSE, -ACF1)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be
treated as missing.
## 5 observations are missing between 2008 Jul and 2008 Nov

## # A tibble: 7 × 4
##   .model .type   RMSE    MAE
##   <chr>  <chr>  <dbl>  <dbl>
## 1 HDT    Test  0.267  0.223
## 2 HLT    Test  0.267  0.221
## 3 HWad   Test  0.0825 0.0628
## 4 HWDad  Test  0.0825 0.0628
## 5 HWDmul Test  0.0818 0.0630
## 6 HWmul  Test  0.0817 0.0630
## 7 SES    Test  0.267  0.223
```

```
# Table for MAE and MSE results:
stp1 <- tribble(
  ~step_ahead, ~model,  ~RMSE, ~MAE,
  1, "HDT", 0.16817331, 0.10420211,
  1, "HLT", 0.17319598, 0.11038657,
  1, "MHW", 0.07301154, 0.05553292,
  1, "SES", 0.16823467, 0.10440352,
```

```
  1, "HWad", 0.07308176, 0.05522043,
  1, "HWDad", 0.07308176, 0.05522043,
  1, "HWDmul", 0.07086046, 0.05388931
)
stp4 <- tribble(
  ~step_ahead, ~model, ~RMSE, ~MAE,
  4, "HDT", 0.24012898, 0.18581179,
  4, "HLT", 0.24156500, 0.18599220,
  4, "MHW", 0.07865661, 0.06057076,
  4, "SES", 0.24001809, 0.18604168,
  4, "HWad", 0.07892714, 0.06022092,
  4, "HWDad", 0.07892714, 0.06022092,
  4, "HWDmul", 0.07819006, 0.06010534
)
stp6 <- tribble(
  ~step_ahead, ~model,~RMSE, ~MAE,
  6, "HDT", 0.26710797, 0.22278858,
  6, "HLT", 0.26735159, 0.22100399,
  6, "MHW", 0.08170965, 0.06298554,
  6, "SES", 0.26688740, 0.22280720,
  6, "HWad", 0.08250761, 0.06276994,
  6, "HWDad", 0.08250761, 0.06276994,
  6, "HWDmul", 0.08182528, 0.06299612
)
# Calculate MSE from RMSE
stp1 <- stp1 |>
  mutate(MSE = RMSE^2)

stp4 <- stp4 |>
  mutate(MSE = RMSE^2)

stp6 <- stp6 |>
  mutate(MSE = RMSE^2)

# Combine the data frames
table <- bind_rows(stp1, stp4, stp6) |>
  select(-RMSE)
table <- table |>
  arrange(step_ahead, MAE)

table
```

```
## # A tibble: 21 × 4
##    step_ahead model     MAE     MSE
##         <dbl> <chr>   <dbl>   <dbl>
##  1          1 HWDmul 0.0539 0.00502
##  2          1 HWad   0.0552 0.00534
##  3          1 HWDad  0.0552 0.00534
##  4          1 HWmul  0.0555 0.00533
##  5          1 HDT    0.104  0.0283
##  6          1 SES    0.104  0.0283
##  7          1 HLT    0.110  0.0300
##  8          4 HWDmul 0.0601 0.00611
```

```
##  9          4 HWad   0.0602 0.00623
## 10          4 HWDad  0.0602 0.00623
## 11          4 HWmul  0.0606 0.00619
## 12          4 HDT    0.1858 0.05766
## 13          4 HLT    0.1860 0.05835
## 14          4 SES    0.1860 0.05761
## 15          6 HWad   0.0628 0.00681
## 16          6 HWDad  0.0628 0.00681
## 17          6 HWmul  0.0630 0.00668
## 18          6 HWDmul 0.0630 0.00670
## 19          6 HLT    0.2210 0.07148
## 20          6 HDT    0.2228 0.07135
## 21          6 SES    0.2228 0.07123
```

The analysis of forecast accuracy shows distinct patterns across various forecast horizons and evaluation metrics. Among the examined methods, the Holt-Winters model with damped trend and multiplicative seasonality ("HWDmul") stands out as the top performer. It achieves the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE) values for 1 and 4 step-ahead forecast horizons. Specifically, for 1 step-ahead, it records an MAE of 0.0539 and an MSE of 0.00502, while for 4 step-ahead, the MAE is 0.0601, and the MSE is 0.00611.

However, for the 6 step-ahead forecast, the model with the lowest error metrics is Holt-Winters Additive ("HWad"), with an MAE of 0.0628 and an MSE of 0.0068, closely followed by Holt-Winters with damped trend and additive seasonality.

The models lacking trend and/or seasonality exhibited the poorest accuracy. Furthermore, as the forecast horizon extends, there is a noticeable increase in forecast error, emphasizing the importance of selecting the appropriate step-ahead forecasts to determine the reliability of predictions further into the future.

In summary, the choice of the most accurate forecasting method depends on specific accuracy metric values, with lower errors indicating higher accuracy. The "HWDmul" method demonstrates robust performance for 1-step-ahead and 4-step-ahead forecasts, while "HWad" or "HWDad" may be more suitable for extended forecast horizons. This exceptional performance can be attributed to the model's effectiveness in capturing the underlying seasonality and trend components within the time series data.

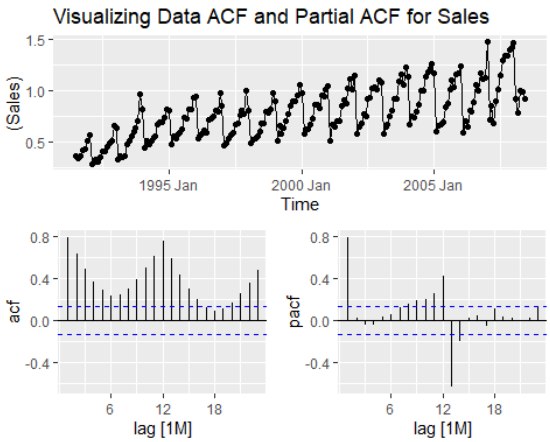
(d) Answer:

Potential errors may arise when selecting a forecasting model that is not suitable for the time series, especially if the data exhibits trends or seasonal patterns. Comparing forecasting methods based solely on metrics like MAE and MSE, without considering model complexity and the risk of overfitting, can result in the choice of overly complex models that perform well on historical data but may not generalize effectively to future data. It's crucial to consider the data's characteristics and forecasting needs to avoid these errors.

Question 2

(a) Answer:

```
# Autocorrelation plot:
df |>
  gg_tsdisplay((Sales), plot_type='partial') +
  labs(title="Visualizing Data ACF and Partial ACF for Sales")
```

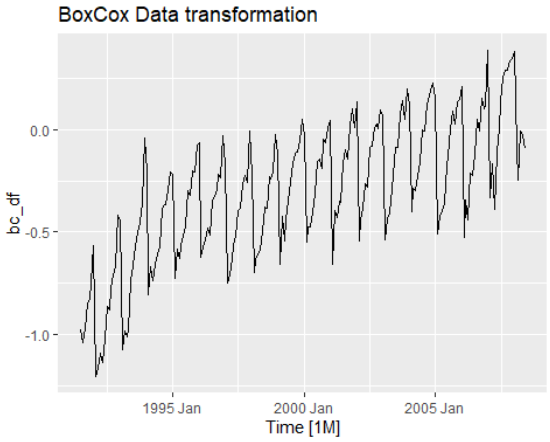


a1 - Upon analyzing the Autocorrelation Function (ACF) plot, it becomes evident that seasonality is a prominent feature within the time series data. The ACF plot exhibits a notable initial spike at lag 1, signifying a strong autocorrelation. Furthermore, the ACF plot demonstrates a gradual decline up to lag 6 and 7, followed by a slow increase. This observed behavior in the ACF plot provides strong evidence that the time series is non-stationary, supporting our earlier conclusion in response to question 1.

a2 - Indeed, applying transformation and differencing to the series is a suitable approach. These techniques assist to stabilize the mean of the time series by eliminating variations in its level, effectively mitigating both trend and seasonality effects.

(b) Answer:

```
# Applying Box-Cox transformation:
lambda <- df |>
features(Sales, features = guerrero) |>
pull(lambda_guerrero)
bc_df <- df |>
  mutate(bc_df = box_cox(Sales, lambda = lambda))
bc_df <- bc_df[, -2]
autoplot(bc_df) +
  labs(title="BoxCox Data transformation")

## Plot variable not specified, automatically selected `.vars = bc_df`
```



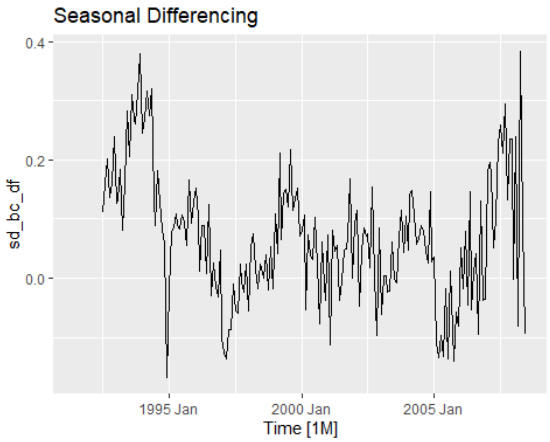
```
head(bc_df)
```

```
## # A tsibble: 6 x 2 [1M]
##       Time  bc_df
##      <mth>  <dbl>
## 1 1991 Jul -0.977
## 2 1991 Aug -1.04
## 3 1991 Sep -0.972
## 4 1991 Oct -0.848
## 5 1991 Nov -0.830
## 6 1991 Dec -0.656
```

While the variance seems to have stabilized, there are still observable indications of both trend and seasonality in the data. To address this, applying seasonal difference can be a valuable step in removing the trend component and further enhancing the stationarity of the series.

```
# Applying  seasonal difference:
sd_bc_df <- bc_df |>
  mutate(sd_bc_df = difference(bc_df, lag=12))
sd_bc_df |>
  autoplot(sd_bc_df) +
  labs(title="Seasonal Differencing")
```

```
## Warning: Removed 12 row(s) containing missing values (geom_path).
```



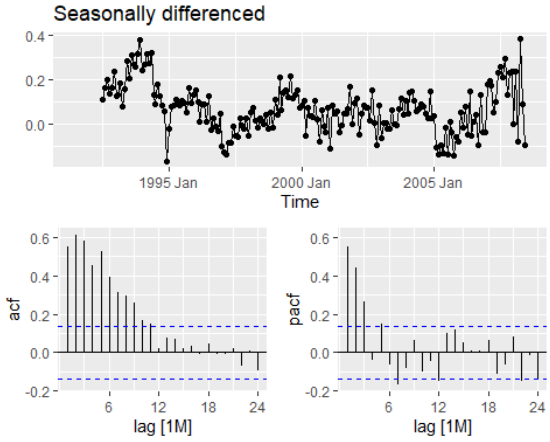
The seasonal differencing plot reveals an absence of trend in the data. Nevertheless, the variance does not center around zero. We should investigate whether additional seasonal differencing is required. It seems that an additional seasonal difference is not needed, as indicated by the unit root test for seasonal differencing.

```
# check if it is necessary additional seasonal difference:
sd_bc_df |>
features(sd_bc_df, unitroot_nsdiffs)
```

```
## # A tibble: 1 × 1
##   nsdiffs
##     <int>
## 1       0
```

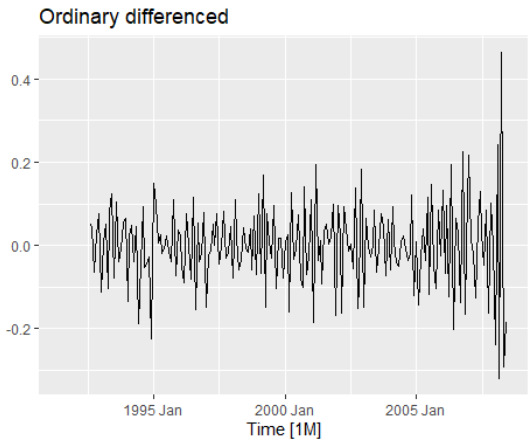
erve if the data still non-staionary by ACF plot.

```
sd_bc_df |>
  gg_tsdisplay(sd_bc_df, plot_type = 'partial', lag = 24) +
      labs(title="Seasonally differenced", y= "")

## Warning: Removed 12 row(s) containing missing values (geom_path).

## Warning: Removed 12 rows containing missing values (geom_point).
```



Apparently by observing the ACF the data still non-stationary. Therefore, we take a further ordinary differenced.

```
od_sd_bc_df <- sd_bc_df |>
  mutate(od_sd_bc_df = difference(sd_bc_df))
od_sd_bc_df |>
  autoplot(od_sd_bc_df) +
  labs(title="Ordinary differenced", y= "")

## Warning: Removed 13 row(s) containing missing values (geom_path).
```




```
# Check if additional ordinary difference is necessary:
od_sd_bc_df |>
  features(od_sd_bc_df, unitroot_ndiffs)
```

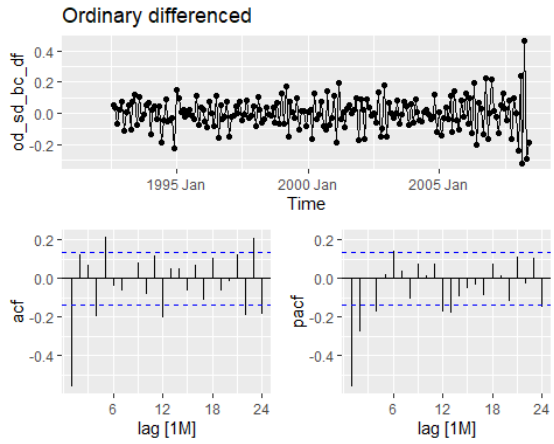
```
## # A tibble: 1 × 1
##   ndiffs
##    <int>
## 1      0
```

After applying the first-order differencing, it becomes evident that the data is close to stationary, as it floats around a mean of zero in the plot. Additionally, a check using the unit root test (ndiffs) to determine if further differencing is necessary confirms that it is not needed. Therefore, we can confidently conclude that d = 1 and D = 1.

Question 3

(a) Answer:

```
# Applying ACF to analyse the white noise, and stationarity
od_sd_bc_df |>
  gg_tsdisplay(od_sd_bc_df, plot_type='partial', lag = 24) +
  labs(title="Ordinary differenced")

## Warning: Removed 13 row(s) containing missing values (geom_path).

## Warning: Removed 13 rows containing missing values (geom_point).
```



When employing the ARIMA model, we establish the parameters denoted by single letters for both non-seasonal and seasonal ARIMA models. Additionally, we set the value of "m" to 12, signifying that the time series is structured in monthly intervals, with "m" representing the seasonality period in the ARIMA model.

The autocorrelation function (ACF) reveals one small spike at lag 12 and 24, indicating the presence of a seasonal MA(1 or 2) component. Additionally, the partial autocorrelation function (PACF) shows a a small spike at lag 12, suggesting a seasonal AR(1) component.

Regarding the non-seasonal aspect, the PACF showcases significant spikes at lags 1, 2, and 4, suggesting a non-seasonal AR(1, 2, or 3) component. It is imperative to disregard the spike at lag 12 when considering the non-seasonal AR component. Simultaneously, the ACF displays peaks at lags 1, 4, and 5, implying non-seasonal MA(1, 2, or 3) components.

The proposed ARIMA models are: ARIMA(1,1,0)(1,1,1)12, ARIMA(1,1,0)(1,1,0)12, ARIMA(2,1,0)(1,1,1)12, ARIMA(2,1,0)(1,1,0)12, ARIMA(1,1,3)(1,1,1)12, ARIMA(1,1,3)(1,1,0)12, ARIMA(2,1,3)(1,1,1)12, ARIMA(2,1,3)(1,1,2)12, ARIMA(1,1,0)(1,1,2)12, ARIMA(2,1,0)(1,1,2)12, ARIMA(3,1,2)(1,1,1)12,

ARIMA(3,1,2)(1,1,1)12, ARIMA(3,1,0)(1,1,1)12, ARIMA(3,1,0)(1,1,0)12, ARIMA(3,1,3)(1,1,1)12, ARIMA(3,1,3)(1,1,0)12

```
# Lets process the proposed models:
arm_fit <- bc_df |>
    model(
      arima110110 = ARIMA(bc_df ~ 0 + pdq(1,1,0) + PDQ(1,1,1)),
      arima111111 = ARIMA(bc_df ~ 0 + pdq(1,1,0) + PDQ(1,1,0)),
      arima210111 = ARIMA(bc_df ~ 0 + pdq(2,1,0) + PDQ(1,1,1)),
      arima210110 = ARIMA(bc_df ~ 0 + pdq(2,1,0) + PDQ(1,1,0)),
      arima113111 = ARIMA(bc_df ~ 0 + pdq(1,1,3) + PDQ(1,1,1)),
      arima113110 = ARIMA(bc_df ~ 0 + pdq(1,1,3) + PDQ(1,1,0)),
      arima310111 = ARIMA(bc_df ~ 0 + pdq(3,1,0) + PDQ(1,1,1)),
      arima310110 = ARIMA(bc_df ~ 0 + pdq(3,1,0) + PDQ(1,1,0)),
      arima312111 = ARIMA(bc_df ~ 0 + pdq(3,1,2) + PDQ(1,1,1)),
      arima312110 = ARIMA(bc_df ~ 0 + pdq(3,1,2) + PDQ(1,1,0)),
      arima110112 = ARIMA(bc_df ~ 0 + pdq(1,1,0) + PDQ(1,1,2)),
      arima210112 = ARIMA(bc_df ~ 0 + pdq(2,1,0) + PDQ(1,1,2)),

      stepwise = ARIMA(bc_df),
      search   = ARIMA(bc_df, stepwise =  FALSE)
    )

## Warning in sqrt(diag(best$var.coef)): NaNs produced

arm_fit

## # A mable: 1 x 14
##                 arima110110               arima111111               arima210111
##                     <model>                   <model>                   <model>
## 1 <ARIMA(1,1,0)(1,1,1)[12]> <ARIMA(1,1,0)(1,1,0)[12]> <ARIMA(2,1,0)(1,1,1)[12]>
## # … with 11 more variables: arima210110 <model>, arima113111 <model>,
## #   arima113110 <model>, arima310111 <model>, arima310110 <model>,
## #   arima312111 <model>, arima312110 <model>, arima110112 <model>,
## #   arima210112 <model>, stepwise <model>, search <model>

glance(arm_fit) |> arrange(AICc) |> select(.model:BIC)

## # A tibble: 13 × 6
##    .model       sigma2 log_lik   AIC  AICc   BIC
##    <chr>         <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1 search      0.00461    241. -468. -468. -446.
##  2 stepwise    0.00469    238. -466. -466. -450.
##  3 arima210111 0.00473    238. -465. -465. -449.
##  4 arima210112 0.00471    239. -465. -465. -446.
##  5 arima310111 0.00474    238. -464. -463. -444.
##  6 arima113111 0.00483    237. -460. -460. -437.
##  7 arima110110 0.00520    227. -445. -445. -432.
##  8 arima110112 0.00517    228. -445. -445. -429.
##  9 arima210110 0.00572    222. -437. -437. -424.
## 10 arima312110 0.00562    225. -437. -436. -414.
## 11 arima310110 0.00575    222. -435. -434. -419.
## 12 arima113110 0.00584    222. -431. -431. -412.
## 13 arima111111 0.00656    209. -412. -412. -403.
```

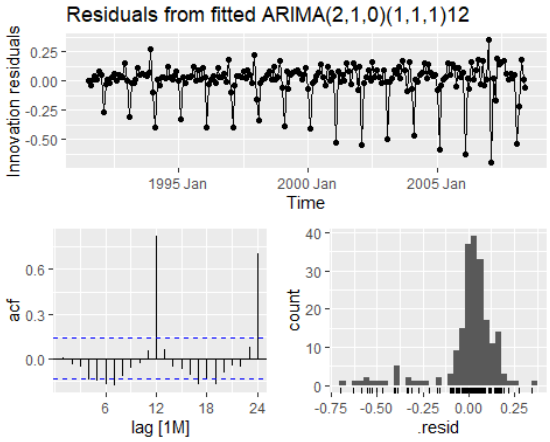
Among the twelve fitted models, the search process identified that ARIMA(2,1,3)(0,1,1)12 had the lowest AICc value at -468. The models initially derived from the ACF and PACF plots were led by ARIMA(2,1,0)(1,1,1)12, which generated an AICc of -465, indicating a close proximity to the suggested ARIMA models from the search process.

(b) Answer:

It is a common practice to include a constant term in time series models when the differencing parameter (d) is either 0 or 1, as this often leads to an improved AICc value. However, when d exceeds 1 (d > 1), it is customary to omit the constant term. In the case of the H03 Australian time series, where d equals 1, adding a constant initially seemed appropriate. However, upon implementation, R issued a warning suggesting reconsideration. The warning stated, "This is generally discouraged; consider removing the constant or reducing the number of differences." It indicated that the model specification might be inducing a quadratic or higher-order polynomial trend. Consequently, the constant was removed, likely due to the presence of a polynomial trend of order d in the forecast.

(c) Answer:

```
# Fitting the ARIMA model from the proposed model with the best AICc
ARIMA(2,1,0)(1,1,1)12 and residuals:
arm_fit <- bc_df |>
  model(
    arima210111 = ARIMA(bc_df ~ pdq(2,1,0) + PDQ(1,1,1))
     )
fit |>
  gg_tsresiduals(lag=24) +
  labs(title="Residuals from fitted ARIMA(2,1,0)(1,1,1)12")
```

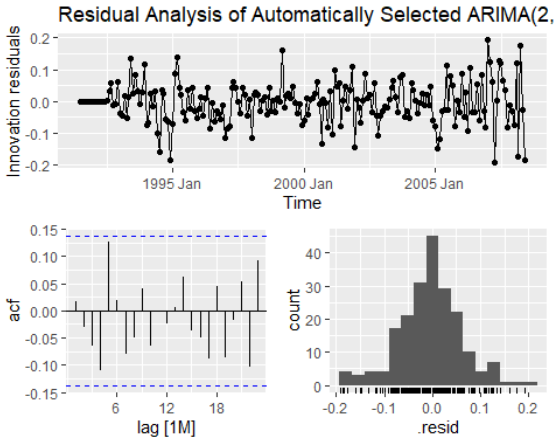


Observing the best proposal ARIMA model, based on ACF, it becomes apparent that there is some autocorrelation present, therefore no white noise. There is a notable spike at lag 12 and 24, accompanied by minor spikes at lags 6, 7, 17, and 19, which marginally surpass the established threshold represented by the blue line. Additionally, there is a noticeable variance change. Consequently, it is apparent that the proposed model falls short of being satisfactory.

(d) Answer:

```
# Use ARIMA_select to automatically select the best ARIMA model
arima_fit <- bc_df |>
  model(ARIMA(bc_df))
```

```
report(arima_fit)

## Series: bc_df
## Model: ARIMA(2,1,0)(0,1,2)[12]
##
## Coefficients:
##           ar1      ar2     sma1     sma2
##       -0.7685  -0.3486  -0.5649  -0.1969
## s.e.   0.0726   0.0729   0.0846   0.0897
##
## sigma^2 estimated as 0.004689:  log likelihood=238.24
## AIC=-466.48   AICc=-466.15   BIC=-450.22

# Residuals:
arima_fit <- bc_df |>
  model(
    arima210012 = ARIMA(bc_df ~ pdq(2,1,0) + PDQ(0,1,2))
     )
arima_fit |>
  gg_tsresiduals() +
  labs(title = "Residual Analysis of Automatically Selected ARIMA(2,1,0)(0,1,2)12
Model")
```



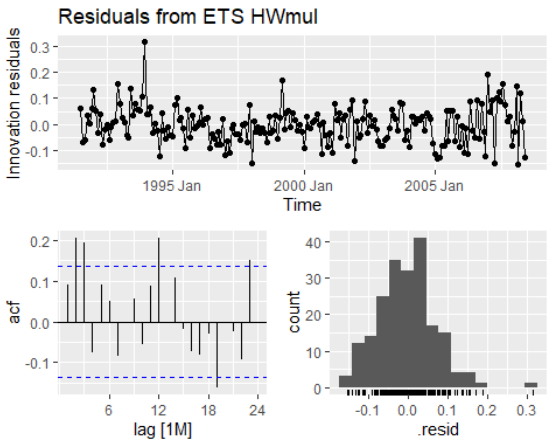
The ARIMA() function model chosen for this data is different and has outperformed the proposed model. The ARIMA model exhibited characteristics such as normal distribution, white noise, and the absence of variance in the data, ultimately resulting in a stationary dataset.

(e) Answer:

Upon examining the residuals from both the ARIMA and ETS models, it becomes evident that ARIMA is the superior choice for forecasting this time series. This conclusion is drawn from the fact that the ARIMA model exhibits residuals with characteristics such as white noise, a normal distribution, and a mean that is closer to zero. In contrast, the ETS model shows deviations from white noise, as evidenced by spikes exceeding the threshold indicated by the blue line, which suggests the presence of non-random patterns in the residuals (see below).

```
# ETS model residuals plot:
fit_ets <- df |>
```

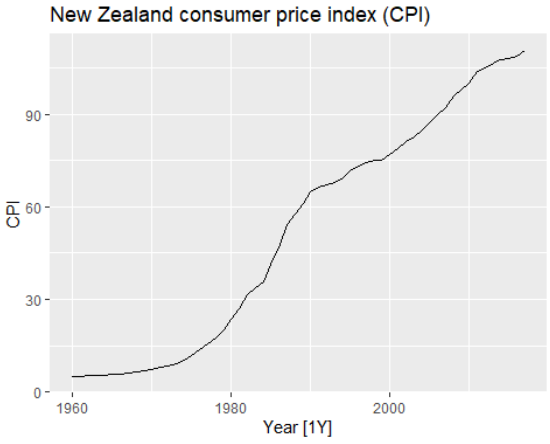
```
  model(Multiplicative = ETS(Sales ~ error("M") + trend("A") + season("M")))
gg_tsresiduals(fit_ets, lag_max = 24) +
  labs(title = "Residuals from ETS HWmul")
```



Question 4

(a) Answer:

```
nz_cpi <- global_economy|>
  filter(Country == "New Zealand")
nz_cpi |>
  autoplot(CPI) +
  labs(title = "New Zealand consumer price index (CPI)")
```




```
# Fitting Holt's linear trend, Holt-Winters additive and a Holt-Winters
Multiplicativce models:
fit_all <- nz_cpi |>
  model(
    HLT = ETS(CPI~ error("A") + trend("A") + season("N")),
```

```
    HWad = ETS(CPI ~ error("A") + trend("A") + season("A")),
    HWmul = ETS(CPI ~ error("M") + trend("A") + season("M")))
```

```
## Warning: 1 error encountered for HWad
## [1] A seasonal ETS model cannot be used for this data.

## Warning: 1 error encountered for HWmul
## [1] A seasonal ETS model cannot be used for this data.
```

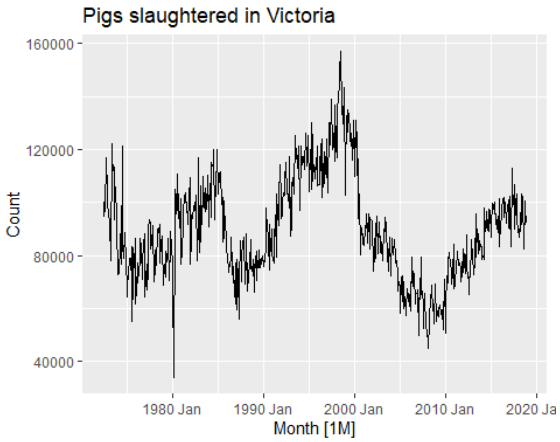
```
accuracy(fit_all)
```

```
## # A tibble: 3 × 11
##   Country    .model .type       ME   RMSE     MAE     MPE   MAPE    MASE   RMSSE
##   <fct>      <chr>  <chr>    <dbl>  <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
## 1 New Zeala… HLT    Trai…   0.0447   1.08   0.759   0.721   1.86   0.409   0.448
## 2 New Zeala… HWad   Trai… NaN      NaN    NaN     NaN     NaN    NaN     NaN
## 3 New Zeala… HWmul  Trai… NaN      NaN    NaN     NaN     NaN    NaN     NaN
## # … with 1 more variable: ACF1 <dbl>
```

The Holt-Winters models return NaN values for accuracy because they are designed to capture seasonality, but the dataset lacks clear seasonality. Given the dataset's linear uptrend and non-seasonal nature, the most suitable model is Holt's Linear, which is specifically fitted for such data.

(b) Answer:

```
# Number of pigs slaughtered in Victoria analysis:
series <- aus_livestock |>
  filter(Animal == "Pigs", State == "Victoria")
series |>
  autoplot(Count) +
  labs(title = "Pigs slaughtered in Victoria")
```




```
# Fitting Holt's linear trend, Holt-Winters additive and a Holt-Winters
Multiplicative models:
fit_series <- series |>
```

```
  model(
    HLT = ETS(Count~ error("A") + trend("A") + season("N")),
    HWad = ETS(Count ~ error("A") + trend("A") + season("A")),
    HWmul = ETS(Count ~ error("M") + trend("A") + season("M"))
)
accuracy(fit_series)
```

```
## # A tibble: 3 × 12
##   Animal State  .model .type    ME  RMSE   MAE    MPE  MAPE  MASE RMSSE     ACF1
##   <fct>  <fct>  <chr>  <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 Pigs   Victo… HLT    Trai…  417. 9386. 7244. -0.430  8.33 0.782 0.755  0.0080
## 2 Pigs   Victo… HWad   Trai…  221. 7755. 5747. -0.274  6.74 0.620 0.624 - 0.0686
## 3 Pigs   Victo… HWmul  Trai…  355. 7794. 5849. -0.144  6.84 0.631 0.627  0.0151
```

Upon analyzing the dataset "aus_livestock," which tracks the number of pigs slaughtered in Victoria, we observe an inconsistent trend. There is a noticeable upward trend from the late 1980s to the late 1990s, followed by a decline towards the end of the 2000s, indicating visible fluctuations. The presence of clear seasonality in the data is not immediately evident.

When we applied ETS models, the training test showed no warnings or NaN values by the Holt-Winters methods. This absence of warnings or NaN values suggests the existence of some level of both seasonality and trend within the dataset. Among the three models considered, the Holt-Winters additive model (HWad) demonstrated the highest accuracy presenting low error metrics.

Rationale: Generally, the prediction intervals produced by ARIMA models expand as the forecast horizon increases. In the case of stationary models, when d is 0, these intervals will converge, making prediction intervals relatively consistent for longer forecast horizons. However, when d is greater than or equal to 1, the prediction intervals will persistently widen as the forecast extends further into the future.

Rationale: The AICc serves as a valuable tool for model selection within the same model class, enabling the choice between candidate ARIMA models or ETS models. However, it is not suitable for comparing ETS and ARIMA models directly because these models belong to distinct model classes, and their likelihoods are computed using different methods.

Rationale: Time series cross-validation can be used to compare between ARIMA and ETS models, especially when dealing with non-seasonal data. In time series cross-validation, the dataset is split into multiple training and testing sets, allowing the models to be trained on one subset of the data and tested on another. By comparing the performance of ARIMA and ETS models on these different subsets, it's possible to assess how well each model class handles the specific characteristics of the time series data.

Rationale: The residual analysis for both models indicated similar white noise patterns. However, when assessing their forecasting performance on the test set, it became apparent that the ARIMA model slightly outperformed the other model, as evidenced by lower RMSE, MAPE, and MASE values.